L Number	Hits	Search Text	DB	Time stamp
1	41	inversion near5 safe	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 20:44
2	58	inheritance near5 priorit$6	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 20:44
3	4	(inheritance near5 priorit$6) and (inversion near5 safe)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 20:58
4	135	pip same variable$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 20:59
5	0	(inheritance near5 priorit$6) and (pip same variable$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 20:59
6	6	(inheritance near5 priorit$6) same variable$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 21:02
7	3	inversion$1 and ((inheritance near5 priorit$6) same variable$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/09/30 21:03

US-PAT-NO: 6560628

DOCUMENT-IDENTIFIER: US 6560628 B1

TITLE: Apparatus, method, and recording medium for scheduling execution using time slot data

DATE-ISSUED: May 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Murata; Seiji	Tokyo	N/A	N/A	JP

US-CL-CURRENT: 709/103, 709/100 , 709/102

ABSTRACT:

A scheduling method for use with a multi-thread system which is capable of time-sharing processing a plurality of threads is provided which can avoid the drawback of priority inversion, minimize the modification of a wait queue, and ensure the optimum use of the processing time of a CPU.

According to the present invention, a time slot data is assigned to each thread and the scheduling is carried out on the basis of the time slot data. As a processing time is imparted to the time slot data, the execution of the thread to which the time slot data is assigned is started. In case that a higher priority thread has to wait for the completion of the execution of a lower priority thread, the time slot data assigned to the higher priority thread is handled as a time slot data of the lower priority thread, hence allowing the execution of the lower priority thread to be started upon a processing time imparted to the time slot data.

12 Claims, 27 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 17

---------- KWIC ---------

Detailed Description Text - DETX (14):

The **variable** "inherit" is used when the priority level of the thread is transferred or inherited from another thread and set to a value indicating the another thread from which the priority level is inherited. In other words, the **variable** "inherit" expresses that the thread of interest is in its wait state to wait for the completion of the execution of the interrupting thread. If there is no **inheritance of the priority** level, the **variable** "inherit" is set to NULL.

Detailed Description Text - DETX (17):

In the thread data, "depend" and "inherit" represent link data to a piece of information related to the **priority inheritance**. For example, when the thread A is in the wait state before the execution of the thread B is completed, the **variable** "depend" in the thread data of the thread A and the **variable** "inherit" in the thread data of the thread B are associated with each other forming a two-directional link. It is noted that the **variable** in the thread data of a

thread is referred to as a **variable** of the thread in the following description. Also, the thread data may be called a thread for ease of the description.

Detailed Description Text - DETX (20):

In the scheduling described below, a **priority inheritance** is in effect conducted by changing a thread specified by a **variable** "owner" and hence by changing a thread for which the processing time of the CPU is imparted. More specifically, when a processing time of the CPU is imparted to the time slot data, the execution of the thread indicated by the **variable** "owner" of the time slot data is started. The **priority inheritance** is thus implemented by modifying the **variable** "owner" instead of directly operating on the wait queue.

US-PAT-NO: 6560628

DOCUMENT-IDENTIFIER: US 6560628 B1

TITLE: Apparatus, method, and recording medium for scheduling execution using time slot data

DATE-ISSUED: May 6, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Murata; Seiji	Tokyo	N/A	N/A	JP

US-CL-CURRENT: 709/103, 709/100 , 709/102

ABSTRACT:

A scheduling method for use with a multi-thread system which is capable of time-sharing processing a plurality of threads is provided which can avoid the drawback of priority **inversion,** minimize the modification of a wait queue, and ensure the optimum use of the processing time of a CPU.

According to the present invention, a time slot data is assigned to each thread and the scheduling is carried out on the basis of the time slot data. As a processing time is imparted to the time slot data, the execution of the thread to which the time slot data is assigned is started. In case that a higher priority thread has to wait for the completion of the execution of a lower priority thread, the time slot data assigned to the higher priority thread is handled as a time slot data of the lower priority thread, hence allowing the execution of the lower priority thread to be started upon a processing time imparted to the time slot data.

12 Claims, 27 Drawing figures

Exemplary Claim Number: 1

Number of Drawing Sheets: 17

---------- KWIC ---------

Abstract Text - ABTX (1):

A scheduling method for use with a multi-thread system which is capable of time-sharing processing a plurality of threads is provided which can avoid the drawback of priority **inversion,** minimize the modification of a wait queue, and ensure the optimum use of the processing time of a CPU.

Brief Summary Text - BSTX (6):

With the use of an algorithm for scheduling in order of the priority, a drawback known as priority **inversion** may arise in the exclusive controlling of access to common resources. Such priority **inversion** is now explained referring to FIG. 1.

Brief Summary Text - BSTX (7):

FIG. 1 is a diagram showing a typical attitude of the priority **inversion**. The priority **inversion** commonly occurs when a higher priority thread A enters a

critical section CS1. It is noted that the critical section CS1 contains a resource commonly used by the thread A and a thread C.

Brief Summary Text - BSTX (9):

Under this condition, when a thread B of which the priority is between those of the thread A and the thread C is turned to its ready state before the thread C leaves from the critical section A, the priority **inversion** occurs. More specifically, even while the higher priority thread A is in its wait state, the execution of the thread B which is lower in the priority than the thread A is started. As the middle priority thread B has been turned to its ready state, the execution of the thread C is interrupted and the execution of the middle priority thread B is started. At the time, the thread A is still in its wait state. Since the relation between the thread A and the thread B is not defined except their priorities, the thread A will hardly presume how long its wait state lasts before its execution is resumed.

Brief Summary Text - BSTX (10):

With the algorithm for scheduling in order of priority, such priority **inversion** may substantially occur causing a higher priority thread to wait for the completion of the execution of a lower priority thread. Also, when the priority **inversion** has once occurred, the duration of interruption of the execution of the higher priority thread will hardly be estimated. Failure of estimating the interruption time for a higher priority thread is critical for a real-time system (that is irresponsible if a process is not finished within a specific length of time).

Brief Summary Text - BSTX (11):

For eliminating the drawback of priority **inversion,** priority inheritance schemes have been introduced (such as "Priority Inheritance Protocols: An Approach to Real-time Synchronization" by Lui Sha, Ragunathan Rajkumar, and John P. Lehoczky, Technical Report CMU-CS-87-181, the Computer Science Department of Carnegie Mellon University, November 1987). The priority inheritance scheme is designed in which when a higher priority thread is turned to its wait state due to the presence of a lower priority thread, the lower priority thread inherits the priority level of the higher priority thread.

Brief Summary Text - BSTX (19):

More particularly, the priority inheritance scheme is effective to eliminate the drawback of priority **inversion,** but demands for dynamically changing the priority level at high frequency thus increasing the overhead due to rescheduling and declining the overall performance of the system.

Brief Summary Text - BSTX (24):

The present invention has been developed in view of the foregoing aspects and its object is to provide an apparatus and a method for scheduling in which the drawback of priority **inversion** can be resolved. Using the scheduling apparatus or method of the present invention, modification of the wait queue which may cause overhead can be minimized and a shared processing time of a CPU can be effectively used. Another object of the present invention is to provide a recording medium in which a data processing program of an operating system capable of time-sharing processing a plurality of executable subjects.

Brief Summary Text - BSTX (35):

According to the present invention, the time slot data assigned to a higher priority thread is used as a time slot data of a lower priority thread thus permitting the priority level of the higher priority thread to be transferred to the lower priority thread. As a result, the drawback of priority **inversion** will be resolved.

Drawing Description Text - DRTX (2):
FIG. 1 is a diagram explaining the drawback of priority **inversion**;

Detailed Description Text - DETX (14):
The **variable** "inherit" is used when the priority level of the thread is transferred or inherited from another thread and set to a value indicating the another thread from which the priority level is inherited. In other words, the **variable** "inherit" expresses that the thread of interest is in its wait state to wait for the completion of the execution of the interrupting thread. If there is no **inheritance of the priority** level, the **variable** "inherit" is set to NULL.

Detailed Description Text - DETX (17):
In the thread data, "depend" and "inherit" represent link data to a piece of information related to the **priority inheritance**. For example, when the thread A is in the wait state before the execution of the thread B is completed, the **variable** "depend" in the thread data of the thread A and the **variable** "inherit" in the thread data of the thread B are associated with each other forming a two-directional link. It is noted that the **variable** in the thread data of a thread is referred to as a **variable** of the thread in the following description. Also, the thread data may be called a thread for ease of the description.

Detailed Description Text - DETX (20):
In the scheduling described below, a **priority inheritance** is in effect conducted by changing a thread specified by a **variable** "owner" and hence by changing a thread for which the processing time of the CPU is imparted. More specifically, when a processing time of the CPU is imparted to the time slot data, the execution of the thread indicated by the **variable** "owner" of the time slot data is started. The **priority inheritance** is thus implemented by modifying the **variable** "owner" instead of directly operating on the wait queue.

Detailed Description Text - DETX (103):
In other words, the scheduling method of the present invention when a higher priority thread has to wait for the completion of the execution of a lower priority thread resolves the drawback of priority **inversion** by handling the lower priority thread instead of the higher priority thread but not by increasing the priority level of the lower priority thread.

Detailed Description Text - DETX (104):
In the scheduling method of the present invention, when the higher priority thread has to wait for the completion of the execution of the lower priority thread, such troublesome operations as modifying the wait queue or recalculating the priority level for priority inheritance are not needed to have the effect that the priority level is transferred from the higher priority thread to the lower priority thread. Accordingly, the scheduling method is advanced to resolve the drawback of priority **inversion** which may be critical in the real-time system. Also, without modifying the wait queue nor recalculating the priority level for priority inheritance, the scheduling method will avoid increase of the cost in eliminating the drawback of priority **inversion**. Moreover, the scheduling method allows the processing time of the CPU imparted to a thread at its wait state to be assigned to an interrupting thread which causes the wait state of the thread, hence ensuring the optimum use of the processing time of the CPU without loss.

Detailed Description Text - DETX (119):

The description of a procedure of switching from one thread to another according to the present invention follows. It is assumed that the first object 34 in the first application 32 sends a process request message to the second object 35. The message is transmitted via the first message handler 23 to the second object 35. The first message handler 23 compares the priority level of the first object 34 as a sender with the priority level of the second object 35 as a receiver to detect an occurrence of priority **inversion**. Upon detecting the priority **inversion,** the first message handler 23 calls the function "makeWait (Thread* target, Thread* depend)". The argument "target" indicates the thread of the first object 23 while the argument "depend" indicates the thread of the second object 35. Accordingly, the time slot data specifying the thread of the first object 34 is transferred to the thread of the second object 35.